AN3980
Application note

STM32™ firmware library for dSPIN L6470

1 Introduction

This application note describes the implementation of the STM32 firmware library for the dSPIN stepper motor control product (L6470). It provides a guide on how to use the library for final application development.
The L6470 is a fully integrated microstepping motor driver with embedded motion engine and SPI communication interface. A low R_{DSON} DMOS power stage is also a part of the IC. The chip supports different operating modes including a 128-microstep resolution. It is based on digital control core surrounded with a number of different peripherals and protections. Chip operating mode, motion profiles, and all the other parameters are memorized in an embedded set of registers. Communication with the IC is done through an integrated 5 MHz SPI periphery in determined data format according to the documentation. See the L6470 datasheet for more details about registers, application commands, etc. The firmware library speeds up the application development process and saves time consumed by register constant definitions and command routine implementation in the microcontroller source code.

Firmware library main features

- Designed for the STM32F1xx microcontroller family
- STEVAL-PCC009V2 + STM32 value line Discovery demonstration boards supported with library configuration (dspin.h) and development tools project files
- Easy portability due to ANSI C standard compliance
  - Only HAL (“Hardware Abstraction Layer”) routines should be modified when used with another platform
  - Simple portability to STM8™ families thanks to similar HAL routines for peripherals - SPI, GPIOs, etc.
- Library contains project folders (files) for development tools:
  - IAR™ - EWARM v5

    J-Link for STEVAL-PCC009V2

    ST-Link for STM32 VL Discovery
  - IAR - EWARM v6

    J-Link for STEVAL-PCC009V2
  - KEIL - µVision® v4.03, v4.20

    ULINK2® for STEVAL-PCC009V2

    ULINK Pro for STEVAL-PCC009V2

    ST-Link for STM32 VL Discovery
  - Raisonance - RIDE v7

    R-Link for STEVAL-PCC009V2

Contents

List of figures

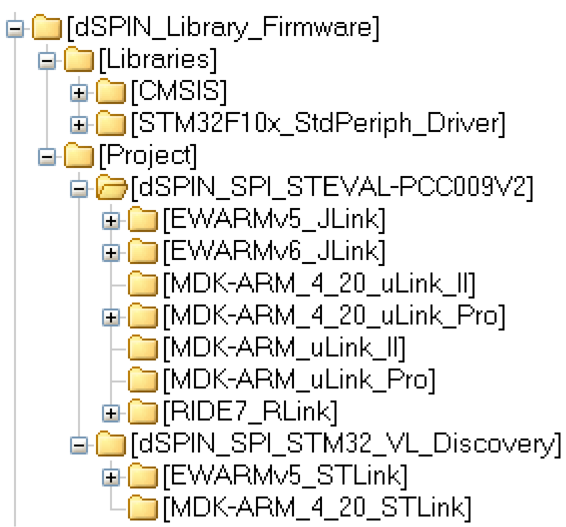
2 Library file structure

Firmware implementation is split between the following files:

- dspin.c
  - Microcontroller peripherals initialization
  - dSPIN application commands implementation
  - Library support functions implementation
- dspin.h
  - Function prototypes for implemented commands and support functions
  - Register value (options) definition
  - Register mask definition
  - Macros for selected function parameter conversions
  - Demonstration board related definitions - GPIO signals and peripherals assignment
- main.c
  - Example of library usage - system configuration / function calls
- Other microcontroller configuration files, such as
  - clock.c, -.h, stm32f10x_conf.h (peripherals configuration), stm32f10x_it.c, -.h (interrupt routines) and standard library for GPIO and SPI

Firmware is available for download in compressed zip format. By decompressing the archive the following folder structure is created on a drive:

Figure 1. Firmware library folder structure



The “Libraries” folder contains microcontroller related files like peripheral source/header files, startup file, etc.

The “Project” folder contains demonstration board related subfolders. dSPIN library source and header files and other subfolders with project files for different development tools appear in each of the demonstration board subfolders. A list of the source/header files can be seen in *Figure 2*:

Figure 2. Firmware library file list

clock	c
clock	h
dspin	c
dspin	h
main	c
main	h
readme	txt
stm32f10x_conf	h
stm32f10x_it	c
stm32f10x_it	h

3 Demonstration board electrical connection

The firmware library supports two demonstration boards:

1. STEVAL-PCC009V2 universal USB to serial bus interface
2. STM32F100 value line Discovery kit

In the case of STEVAL-PCC009V2, the connection of the L6470 demonstration board is simple. It is only necessary to use a 10-wire flat cable connection line inserted into J1 on STEVAL-PCC009V2 and J10 on the L6470 demonstration board. The signals are assigned according to the appropriate section in the dSPIN.h file, see *Table 1*:

Table 1. Microcontroller signal assignment for STEVAL-PCC009V2

L6470 demonstration board signal	Microcontroller periphery
SPI Clock	PB 13
SPI MOSI	PB 15
SPI MISO	PB 14
SPI nSS	PB 12
BUSY	PB 10
FLAG	PB 11
PWM 1	PB 0
PWM 2	PB 1
STEVAL-PCC009V2 Power LED	PC 4

Microcontroller peripherals for the STM32F100 value line Discovery kit connection have been selected according to *Table 2*. The user is free to modify the assignment according to their needs. The condition for correct functionality is consistency between physical connection and dSPIN.h file content.

Table 2. Microcontroller signal assignment for STM32F100 value line Discovery kit

L6470 demonstration board signal	Microcontroller periphery
SPI Clock	PA 5
SPI MOSI	PA 7
SPI MISO	PA 6
SPI nSS	PA 4
BUSY	PC 4
FLAG	PC 5
PWM 1	PA 1
PWM 2	PB 0

4 dSPIN.h file content

The dSPIN.h file contains the following:

- Demonstration board related signal assignments
- Structure definition for L6470 register initialization
- Register options / mask definition (enumerated types)
- L6470 register addresses (enumerated types)
- L6470 command set definition (enumerated types)
- Macro definition - used for function input parameter conversions
- List of function prototypes which are implemented in the dSPIN.c file.

4.1 Demonstration board related signal assignments

Demonstration board type must be defined prior to the signal assignment definition. Therefore, one of the following lines should appear at the beginning of the dSPIN.h file:

Figure 3. Demonstration board selection

```
#define STEVAL_PCC009V2
#define STM32_VL_Discovery
```

It is recommended to keep both in the code and comment the one which does not correspond to the requested configuration.

In *Figure 4* it is possible to see how the signals are assigned depending on the demonstration board type:

Figure 4. Demonstration board signal assignment

```
#ifdef STEVAL_PCC009V2
        #define dSPIN_SPI                       SPI2

        #define dSPIN_SCK_Pin                   GPIO_Pin_13
        #define dSPIN_SCK_Port                  GPIOB

        #define dSPIN_MOSI_Pin                  GPIO_Pin_15
        #define dSPIN_MOSI_Port                 GPIOB

        #define dSPIN_MISO_Pin                  GPIO_Pin_14
        #define dSPIN_MISO_Port                 GPIOB

        #define dSPIN_nSS_Pin                   GPIO_Pin_12
        #define dSPIN_nSS_Port                  GPIOB

        #define dSPIN_BUSY_Pin                  GPIO_Pin_10
        #define dSPIN_BUSY_Port                 GPIOB

        #define dSPIN_FLAG_Pin                  GPIO_Pin_11
        #define dSPIN_FLAG_Port                 GPIOB

        #define dSPIN_PWM1_Pin                  GPIO_Pin_0
        #define dSPIN_PWM1_Port                 GPIOB

        #define dSPIN_PWM2_Pin                  GPIO_Pin_1
        #define dSPIN_PWM2_Port                 GPIOB

        #define POWER_LED_Pin                   GPIO_Pin_4
        #define POWER_LED_Port                  GPIOC

        /* List all the peripherals, which CLKs have to be enabled! */
        #define dSPIN_PERIPHERAL_CLKs_APB1 (RCC_APB1Periph_SPI2)
        #define dSPIN_PERIPHERAL_CLKs_APB2 (RCC_APB2Periph_GPIOB |
RCC_APB2Periph_GPIOC)
#endif
```

The second demonstration board option, STM32 value line Discovery, is implemented in a similar way.

4.2 dSPIN Init structure definition

To configure all L6470 registers a dedicated structure type has been defined:

Figure 5. L6470 registers structure type definition

```
typedef struct
{
  uint32_t ABS_POS;
  uint16_t EL_POS;
  uint32_t MARK;
  uint32_t SPEED;
  uint16_t ACC;
  uint16_t DEC;
  uint16_t MAX_SPEED;
  uint16_t MIN_SPEED;
  uint16_t FS_SPD;
  uint8_t  KVAL_HOLD;
  uint8_t  KVAL_RUN;
  uint8_t  KVAL_ACC;
  uint8_t  KVAL_DEC;
  uint16_t INT_SPD;
  uint8_t  ST_SLP;
  uint8_t  FN_SLP_ACC;
  uint8_t  FN_SLP_DEC;
  uint8_t  K_THERM;
  uint8_t  ADC_OUT;
  uint8_t  OCD_TH;
  uint8_t  STALL_TH;
  uint8_t  STEP_MODE;
  uint8_t  ALARM_EN;
  uint16_t CONFIG;
}dSPIN_RegsStruct_TypeDef;
```

Typically, a variable is created in the main program loop and filled with requested register parameters. Then, the appropriate function call (dSPIN_Registers_Set(&dSPIN_RegsStruct)) can program all registers as requested by the user.

4.3 Register options / mask definition (enumerated types)

As L6470 consists of 25 registers, and many of them include multiple parameter settings, the complete code to cover all the definitions is large. The following lists are just selected examples and the others can be found directly in the header file.

Figure 6. Step select parameters definition for “Step Mode” register

```
typedef enum {
        dSPIN_STEP_SEL_1          =((uint8_t)0x00),
        dSPIN_STEP_SEL_1_2        =((uint8_t)0x01),
        dSPIN_STEP_SEL_1_4        =((uint8_t)0x02),
        dSPIN_STEP_SEL_1_8        =((uint8_t)0x03),
        dSPIN_STEP_SEL_1_16       =((uint8_t)0x04),
        dSPIN_STEP_SEL_1_32       =((uint8_t)0x05),
        dSPIN_STEP_SEL_1_64       =((uint8_t)0x06),
        dSPIN_STEP_SEL_1_128      =((uint8_t)0x07)
} dSPIN_STEP_SEL_TypeDef;
```

Figure 7. Configuration register mask definition

```
typedef enum {
        dSPIN_CONFIG_OSC_SEL      =((uint16_t)0x0007),
        dSPIN_CONFIG_EXT_CLK      =((uint16_t)0x0008),
        dSPIN_CONFIG_SW_MODE      =((uint16_t)0x0010),
        dSPIN_CONFIG_EN_VSCOMP    =((uint16_t)0x0020),
        dSPIN_CONFIG_OC_SD        =((uint16_t)0x0080),
        dSPIN_CONFIG_POW_SR       =((uint16_t)0x0300),
        dSPIN_CONFIG_F_PWM_DEC    =((uint16_t)0x1C00),
        dSPIN_CONFIG_F_PWM_INT    =((uint16_t)0xE000)
} dSPIN_CONFIG_Masks_TypeDef;
```

4.4 L6470 register addresses

To allow access to L6470 registers, the following register address definition has been created.

Figure 8. Register addresses

```
typedef enum {
        dSPIN_ABS_POS              =((uint8_t)0x01),
        dSPIN_EL_POS               =((uint8_t)0x02),
        dSPIN_MARK                 =((uint8_t)0x03),
        dSPIN_SPEED                =((uint8_t)0x04),
        dSPIN_ACC                  =((uint8_t)0x05),
        dSPIN_DEC                  =((uint8_t)0x06),
        dSPIN_MAX_SPEED            =((uint8_t)0x07),
        dSPIN_MIN_SPEED            =((uint8_t)0x08),
        dSPIN_FS_SPD               =((uint8_t)0x15),
        dSPIN_KVAL_HOLD            =((uint8_t)0x09),
        dSPIN_KVAL_RUN             =((uint8_t)0x0A),
        dSPIN_KVAL_ACC             =((uint8_t)0x0B),
        dSPIN_KVAL_DEC             =((uint8_t)0x0C),
        dSPIN_INT_SPD              =((uint8_t)0x0D),
        dSPIN_ST_SLP               =((uint8_t)0x0E),
        dSPIN_FN_SLP_ACC           =((uint8_t)0x0F),
        dSPIN_FN_SLP_DEC           =((uint8_t)0x10),
        dSPIN_K_THERM              =((uint8_t)0x11),
        dSPIN_ADC_OUT              =((uint8_t)0x12),
        dSPIN_OCD_TH               =((uint8_t)0x13),
        dSPIN_STALL_TH             =((uint8_t)0x14),
        dSPIN_STEP_MODE            =((uint8_t)0x16),
        dSPIN_ALARM_EN             =((uint8_t)0x17),
        dSPIN_CONFIG               =((uint8_t)0x18),
        dSPIN_STATUS               =((uint8_t)0x19),
        dSPIN_RESERVED_REG1        =((uint8_t)0x1A),
        dSPIN_RESERVED_REG2        =((uint8_t)0x1B)
} dSPIN_Registers_TypeDef;
```

4.5 L6470 command set definition

dSPIN motion and other types of commands are coded with numeric (single byte) constants. For transparent code implementation, the following type definition has been created:

Figure 9. L6470 command set type definition

```
typedef enum {
        dSPIN_NOP                  =((uint8_t)0x00),
        dSPIN_SET_PARAM            =((uint8_t)0x00),
        dSPIN_GET_PARAM            =((uint8_t)0x20),
        dSPIN_RUN                  =((uint8_t)0x50),
        dSPIN_STEP_CLOCK           =((uint8_t)0x58),
        dSPIN_MOVE                 =((uint8_t)0x40),
        dSPIN_GO_TO                =((uint8_t)0x60),
        dSPIN_GO_TO_DIR            =((uint8_t)0x68),
        dSPIN_GO_UNTIL             =((uint8_t)0x82),
        dSPIN_RELEASE_SW           =((uint8_t)0x92),
        dSPIN_GO_HOME              =((uint8_t)0x70),
        dSPIN_GO_MARK              =((uint8_t)0x78),
        dSPIN_RESET_POS            =((uint8_t)0xD8),
        dSPIN_RESET_DEVICE         =((uint8_t)0xC0),
        dSPIN_SOFT_STOP            =((uint8_t)0xB0),
        dSPIN_HARD_STOP            =((uint8_t)0xB8),
        dSPIN_SOFT_HIZ             =((uint8_t)0xA0),
        dSPIN_HARD_HIZ             =((uint8_t)0xA8),
        dSPIN_GET_STATUS           =((uint8_t)0xD0),
        dSPIN_RESERVED_CMD1        =((uint8_t)0xEB),
        dSPIN_RESERVED_CMD2        =((uint8_t)0xF8)
} dSPIN_Commands_TypeDef;
```

4.6 Macros definition

Some of the L6470 commands use input parameters which are proportional to a real number. For this reason there is a need to recalculate (convert) the requested option parameter in real numeric format to the function parameter in integer format. To make it user friendly and allow usage of input parameters in real numeric format, macros for parameter conversion have been implemented.

For example, the “Run” command supposes to get a speed parameter in unsigned fixed point 0.28 number format. In this case, the “Speed_Steps_to_Par(speed)” macro can be used to convert real speed (in steps per second) to the parameter of the “Run” command.

Figure 10. List of defined macros

```
#define Speed_Steps_to_Par(steps) ((uint32_t)(((steps)*67.108864)+0.5))
/* Speed conversion, range 0 to 15625 steps/s */

#define AccDec_Steps_to_Par(steps) ((uint16_t)(((steps)*0.068719476736)+0.5))
/* Acc/Dec rates conversion, range 14.55 to 59590 steps/s2 */

#define MaxSpd_Steps_to_Par(steps) ((uint16_t)(((steps)*0.065536)+0.5))
/* Max Speed conversion, range 15.25 to 15610 steps/s */

#define MinSpd_Steps_to_Par(steps) ((uint16_t)(((steps)*4.194304)+0.5))
/* Min Speed conversion, range 0 to 976.3 steps/s */

#define FSSpd_Steps_to_Par(steps) ((uint16_t)((steps)*0.065536))
/* Full Step Speed conversion, range 7.63 to 15625 steps/s */

#define IntSpd_Steps_to_Par(steps) ((uint16_t)(((steps)*4.194304)+0.5))
/* Intersect Speed conversion, range 0 to 3906 steps/s */

#define Kval_Perc_to_Par(perc) ((uint8_t)(((perc)/0.390625)+0.5))
/* KVAL conversions, range 0.4% to 99.6% */

#define BEMF_Slope_Perc_to_Par(perc) ((uint8_t)(((perc)/0.00156862745098)+0.5))
/* BEMF compensation slopes, range 0 to 0.4% s/step */

#define KTherm_to_Par(KTherm) ((uint8_t)(((KTherm - 1)/0.03125)+0.5))
/* K_THERM compensation conversion, range 1 to 1.46875 */

#define StallTh_to_Par(StallTh) ((uint8_t)(((StallTh - 31.25)/31.25)+0.5))
/* Stall Threshold conversion, range 31.25mA to 4000mA */
```

5 dSPIN.c file content

dSPIN.c is the library source file. It contains all the L6470 commands and the implementation of other support functions.

It starts with the dSPIN_Peripherals_Init function which configures all the required microcontroller peripherals (GPIOs, SPI) and it is called at the beginning of the main program loop. The function considers signal assignments made in the dSPIN.h file.

The dSPIN_Regs_Struct_Reset function fills all the structure fields with default values.

The dSPIN_Registers_Set function programs all parameters into L6470 internal registers. It uses pointer-to-structure as the input parameter, so the structure must be filled properly in advance of this function call.

The dSPIN.c file contains all the dSPIN commands and the implementation of other support functions. The code is well commented, please refer directly to the file for more details.

6 main.c file content

The main.c file starts with the system clock configuration in order to switch from internal RC oscillator to external crystal circuit. It sets SYSCLK frequency to 24 MHz.

Then, the above mentioned dSPIN_Peripherals_Init function is processed.

The dSPIN_Regs_Struct_Reset function is called to fill all dSPIN_RegsStruct structure fields with default values. As a next step, the majority of the dSPIN_RegsStruct fields are modified either directly or with the help of the macros already described (see *Section 4*). L6470 internal registers are then programmed by the dSPIN_Registers_Set function.

The rest of the main.c file code demonstrates how to use the library functions.

Appendix A Demonstration boards images

Figure 11. Demonstration board STEVAL-PCC009V2

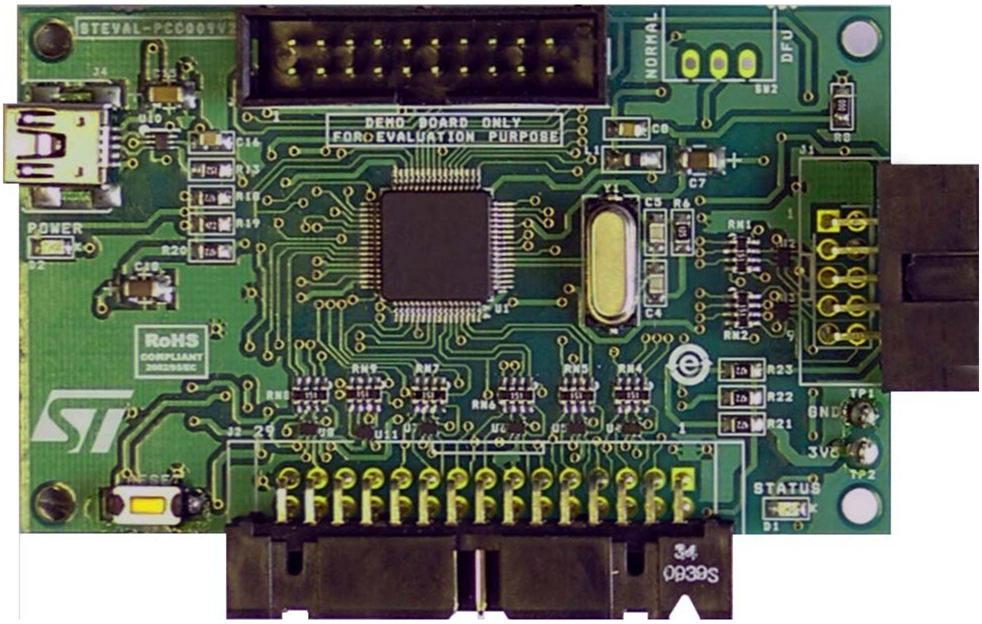


Figure 12. Demonstration board STM32F100 value line Discovery kit

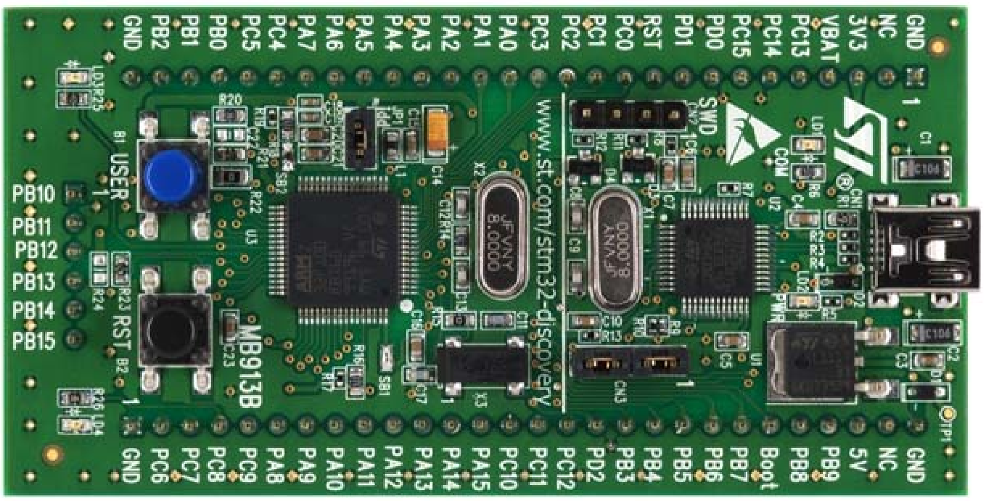
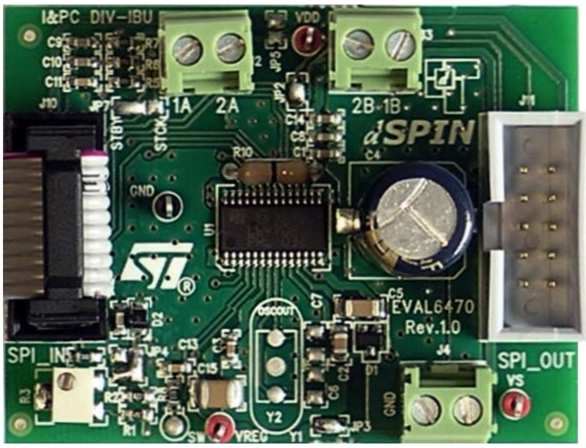


Figure 13. L6470 demonstration board (EVAL6470)



References

1. L6470 datasheet
2. AN3103 application note
3. STM32F100RB datasheet
4. STM32F103RB datasheet
5. RM0041 - STM32F100 reference manual
6. RM0008 - STM32F103 reference manual
7. STEVAL-PCC009V2 databrief
8. UM0919 user manual

Revision history

Table 3. Document revision history

Date	Revision	Changes
22-Nov-2011	1	Initial release.